Task 1

You are given an array of integers a and two integers l and r. You task is to calculate a boolean array b, where b[i] = true if there exists an integer x, such that a[i] = (i + 1) \* x and l ≤ x ≤ r. Otherwise, b[i] should be set to false.

Example

For a = [8, 5, 6, 16, 5], l = 1, and r = 3, the output should be boundedRatio(a, l, r) = [false, false, true, false, true].

* For a[0] = 8, we need to find a value of x such that 1 \* x = 8, but the only value that would work is x = 8 which doesn't satisfy the boundaries 1 ≤ x ≤ 3, so b[0] = false.
* For a[1] = 5, we need to find a value of x such that 2 \* x = 5, but there is no integer value that would satisfy this equation, so b[1] = false.
* For a[2] = 6, we can choose x = 2 because 3 \* 2 = 6 and 1 ≤ 2 ≤ 3, so b[2] = true.
* For a[3] = 16, there is no an integer 1 ≤ x ≤ 3, such that 4 \* x = 16, so b[3] = false.
* For a[4] = 5, we can choose x = 1 because 5 \* 1 = 5 and 1 ≤ 1 ≤ 3, so b[4] = true.

Input/Output

* **[execution time limit] 3 seconds (cs)**
* **[input] array.integer a**

An array of integers.

*Guaranteed constraints:*  
1 ≤ a.length ≤ 100,  
1 ≤ a[i] ≤ 106.

* **[input] integer l**

An integer representing the lower bound for x.

*Guaranteed constraints:*  
1 ≤ l ≤ 104.

* **[input] integer r**

An integer representing the upper bound for x.

*Guaranteed constraints:*  
1 ≤ r ≤ 104,  
l ≤ r.

* **[output] array.boolean**

A boolean array.

Task 2

You are given an array of integers a. A new array b is generated by rearranging the elements of a in the following way:

* b[0] is equal to a[0];
* b[1] is equal to the last element of a;
* b[2] is equal to a[1];
* b[3] is equal to the second-last element of a;
* b[4] is equal to a[2];
* b[5] is equal to the third-last element of a;
* and so on.

Your task is to determine whether the new array b is sorted in **strictly ascending** order or not.

Here is how the process of generating the new array b works:

Example

* For a = [1, 3, 5, 6, 4, 2], the output should be alternatingSort(a) = true.

The new array b will look like [1, 2, 3, 4, 5, 6], which is in strictly ascending order, so the answer is true.

* For a = [1, 4, 5, 6, 3], the output should be alternatingSort(a) = false.

The new array b will look like [1, 3, 4, 6, 5], which is not in strictly ascending order, so the answer is false.

Input/Output

* **[execution time limit] 3 seconds (cs)**
* **[input] array.integer a**

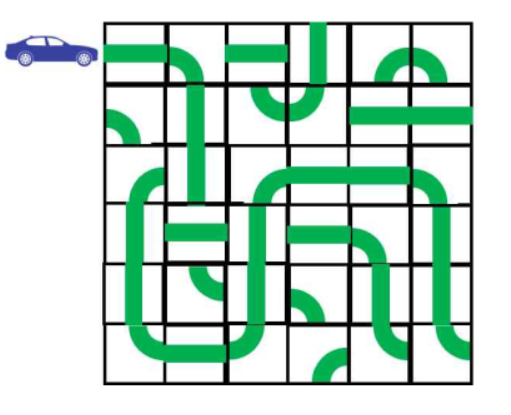
The given array of integers.

*Guaranteed constraints:*  
1 ≤ a.length ≤ 105,  
-109 ≤ a[i] ≤ 109.

* **[output] boolean**

A boolean representing whether the new array b will be sorted in strictly ascending order or not.

Task 3



Consider a map of city streets, in the form of a grid

You'd like to know if it's possible to make your way to the exit, under the following rules:

* You begin from the left side of the square in the top-left corner;
* The exit is on the right side of the square in the bottom-right corner;
* You must travel along a connected path between squares.

You're given directions, a matrix of integers representing the grid of streets, where each integer corresponds to a different type of road square:

* 0 stands for



* 1 stands for



* 2 stands for



* 3 stands for



* 4 stands for



* 5 stands for

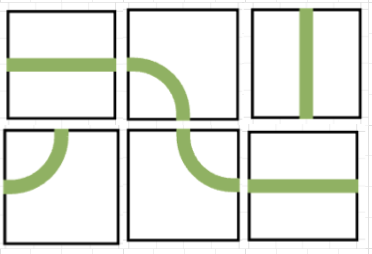


Your task is to return true if it's possible to reach the exit, and false otherwise.

Example

* For directions = [[0, 2, 1], [5, 4, 0]], the output should be trafficMap(directions) = true.

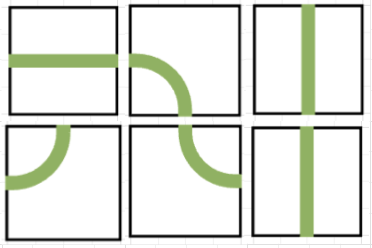
The map looks as follows:



It's possible to enter the top-left square from the left, travel along a connected path, and exit the right side of the bottom-right square. So the answer is true.

* For directions = [[0, 2, 1], [5, 4, 1]], the output should be trafficMap(directions) = false.

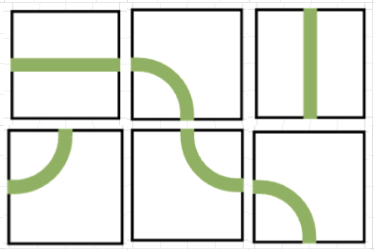
The map looks as follows:



It's possible to enter the top-left square from the left, but there's no connected path that leads to the bottom-right square. So the answer is false.

* For directions = [[0, 2, 1], [5, 4, 2]], the output should be trafficMap(directions) = false.

The map looks as follows:



The path leading to the bottom-right square exists, but it doesn't exit to the right.

* For directions = [[1], [4]], the output should be trafficMap(directions) = false.

The map looks as follows:



It's possible to travel along the path to the exit, but the entrance isn't in the right place.

Input/Output

* **[execution time limit] 3 seconds (cs)**
* **[input] array.array.integer directions**

The map of streets, represented as the rectangular matrix of integers.

*Guaranteed constraints:*  
1 ≤ directions.length ≤ 100,  
1 ≤ directions[0].length ≤ 100,  
directions[i].length = directions[0].length,  
0 ≤ directions[i][j] ≤ 5.

* **[output] boolean**

true if it's possible to reach the right side of the bottom right corner along a connected path from the left side of the top left corner, and false otherwise

Task 4

Given an array of integers a of even length, your task is to split it into two arrays of equal length such that all the numbers are unique in each of them.

There may be more than one possible answer, in which case you may return any of them. If there are no possible answers, return an empty array.

*Hint: Count the number of occurrences of each integer in a. If there are integers occurring more than twice, then there is no solution. Next, put the integers occurring twice into both answer arrays. Finally, put all other numbers in the answer arrays, following the condition that they should have equal sizes.*

Example

* For a = [2, 1, 2, 3, 3, 4], the output can be divideArray(a) = [[2, 1, 3], [2, 3, 4]].

Answers like [[1, 2, 3], [2, 3, 4]] or [[4, 2, 3], [3, 2, 1]] would also be considered correct.

* For a = [1, 2, 2, 1], the output can be divideArray(a) = [[1, 2], [2, 1]].

Again, there are other possible answers.

* For a = [2, 2, 3, 3, 2, 2], the output should be divideArray(a) = [].

No matter how we try to split this array, there will be at least two 2s in at least one of the resulting arrays. So the answer is [].

Input/Output

* **[execution time limit] 3 seconds (cs)**
* **[input] array.integer a**

An array of integers. It is guaranteed that a has even length.

*Guaranteed constraints:*  
2 ≤ a.length ≤ 104,  
1 ≤ a[i] ≤ 105.

* **[output] array.array.integer**

Return an empty array if there is no solution. If a solution exists, return an array of two arrays - a distribution of a where each of these two arrays are of equal length and each contains unique elements.